Fast Algorithm for Finding Unicast Capacity of Linear Deterministic Wireless Relay Networks

Cuizhu Shi and Aditya Ramamoorthy
Department of Electrical and Computer Engineering, Iowa State University,
Ames, Iowa 50011
Email: {cshi, adityar}@iastate.edu

Abstract

The deterministic channel model for wireless relay networks proposed by Avestimehr, Diggavi and Tse '07 has captured the broadcast and inference nature of wireless communications and has been widely used in approximating the capacity of wireless relay networks. The authors generalized the max-flow min-cut theorem to the linear deterministic wireless relay networks and characterized the unicast capacity of such deterministic network as the minimum rank of all the binary adjacency matrices describing source-destination cuts whose number grows exponentially with the size of the network. In this paper, we developed a fast algorithm for finding the unicast capacity of a linear deterministic wireless relay network by finding the maximum number of linearly independent paths using the idea of path augmentation. We developed a modified depth-first search algorithm tailored for the linear deterministic relay networks for finding linearly independent paths whose total number proved to equal the unicast capacity of the underlying network. The result of our algorithm suggests a capacity-achieving transmission strategy with one-bit length linear encoding at the relay nodes in the concerned linear deterministic wireless relay network. The correctness of our algorithm for universal cases is given by our proof in the paper. Moreover, our algorithm has a computational complexity bounded by $O(|\mathcal{V}_x| \cdot C^4 + d \cdot |\mathcal{V}_x| \cdot C^3)$ which shows a significant improvement over the previous results for solving the same problem by Amaudruz and Fragouli (whose complexity is bounded by $O(M \cdot |\mathcal{E}| \cdot C^5)$ with $M \geq d$ and $|\mathcal{E}| \geq |\mathcal{V}_x|$) and by Yazdi and Savari (whose complexity is bounded by $O(L^8 \cdot M^{12} \cdot h_0^3 + L \cdot M^6 \cdot C \cdot h_0^4)$ with $h_0 \geq C$).

I. Introduction

The complex signal interactions in wireless relay networks challenge the study of wireless information flow for many years. To characterize the capacity and capacity-achieving transmission schemes for wireless relay networks still remain open questions. Towards this end,

the deterministic channel model for wireless relay networks proposed by Avestimehr, Diggavi and Tse [1] has been a significant progress. The broadcast and inference are two fundamental features of wireless communications. The deterministic channel model captures the broadcast and inference features of wireless communications in addition to converting the wireless relay networks into deterministic networks. Studying the information flow in the deterministic networks provides a way to find out the approximated capacity and corresponding transmission strategies for original wireless relay networks.

Gaussian channel has been the most widely used channel model for the link channels in wireless relay networks. The deterministic channel model quantizes the transmitted signal into different bit levels and at the receiver keeps the signal bit levels above the noise level (which depends on the signal to noise ratio (SNR) of the channel) so as to convert the original Gaussian channel into a deterministic channel without random noise variables. The broadcasting of signal at the transmitter is still preserved in the deterministic channel and the interference of signal at the receiver is modeled by modulo two sum of the bits arrived at the same signal level.

Now we introduce the deterministic channel model by using the example of a point-to-point AWGN channel from [1]. Consider an AWGN channel $y = hx + z$ where $z \sim \mathcal{N}(0, 1)$ ($\mathcal{N}$ represents Gaussian distribution) and $h = \sqrt{\text{SNR}}$. Assume x and z are real numbers, then we can write $y \approx 2^n \sum_{i=1}^{n} x(i)2^{-i} + \sum_{i=1}^{\infty}(x(i+n) + z(i))2^{-i}$ where $n = \lceil \frac{1}{2} \log \text{SNR} \rceil$. If we think of the transmitted signal x as a sequence of bits at different signal levels, then the deterministic channel model truncates x and passes only its bits above noise level. Fig. 1 gives a concrete example. At the transmitter node T_x and receiver node R_x, each small cycle represents a signal level. Assume $n = 4$, so only the first four most significant signal levels or bits from T_x are received at R_x. Accordingly each edge in the model can transmit one-bit information at a time.

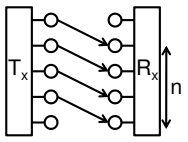

Fig. 1. An example of the deterministic channel model for a point-to-point Gaussian channel.

The deterministic channel model we discussed above is called linear finite-field deterministic channel model in [2], which is referred to as linear deterministic channel model in this paper.

In [1][2], the unicast (i.e., with one source S and one destination D) capacity C of the linear deterministic wireless relay networks was characterized as the minimum rank of all the binary adjacency matrices describing $S-D$ cuts. Since the total number of such cuts grows exponentially with the size of the network, an exhaustive search for the minimum rank of the adjacency matrix for these cuts results in an algorithm with complexity exponential in the size of the network. A polynomial-time algorithm is desirable for finding the unicast capacity of the linear deterministic wireless relay networks. We will see later that the path augmentation algorithms for graphs cannot be directly applied here because the definitions for the cut value for two cases are different (see the definition for the cut value in a linear deterministic wireless relay network in Section II). In this paper, we only consider linear deterministic wireless relay networks. Since an arbitrary deterministic wireless relay network can be unfolded over time to create a layered deterministic network through time-expansion technique [2], our algorithm developed in this paper for layered networks also applies to general networks.

In [3], Amaudruz and Fragouli proposed a polynomial-time algorithm for finding the unicast capacity of a linear deterministic wireless relay network by trying to identify the maximum number of linearly independent paths in the network using the idea of path augmentation. In [4], Yazdi and Savari developed a two-dimensional Rado-Hall transversal theorem for block matrices and used the submodularity of the capacity of a cut to formulate the problem as a linear program over the intersection of two polymatroids so as to solve the problem in polynomial time. Compared with these previous results, our algorithm has a significantly less computational complexity as explained in Section IV-A. Moreover, our algorithm comes with a more intuitive understanding as we explained in Section III.

The paper is organized as follows. In Section II, we introduce the notations and definitions used throughout this paper. Section III gives a detailed description of our algorithm for finding the unicast capacity of any given linear deterministic wireless relay network. Section IV is about the algorithm analysis including the proof of correctness and complexity analysis. Section V concludes the paper.

II. Notations and Definitions

Let $\mathcal{G} = (\mathcal{N}, \mathcal{V}, \mathcal{E})$ be a layered deterministic wireless relay network [2] where $\mathcal{N}$ is the set of super nodes referring to the nodes in the original wireless relay network, $\mathcal{V}$ is the set of nodes

referring to different signal levels incident to super nodes and $\mathcal{E}$ is the set of directed edges going from node to node. Denote $\mathcal{V} = \mathcal{V}_x \cup \mathcal{V}_y$ with $\mathcal{V}_x = \{x : (x, y) \in \mathcal{E}\}$ and $\mathcal{V}_y = \{y : (x, y) \in \mathcal{E}\}$. We shall call $\mathcal{V}_x$ as transmitting nodes and $\mathcal{V}_y$ as receiving nodes. For example, in Fig. 1, $\mathcal{N} = \{T_x, R_x\}$ and T_x has five transmitting nodes and R_x has five receiving nodes. In a layered network, all paths from the source to the destination have equal lengths [2], so we can divide the set of super nodes into different layers each layer containing only super nodes with the same distance to the source. Assume there are L layers of super nodes in $\mathcal{G}$ and M is the maximum number of super nodes in each layer. The source super node S consists the first layer and the destination super node D consists the last layer. Let $\mathcal{N}(x_i)$ (or $\mathcal{N}(y_j)$) denote the super node where a transmitting node x_i (or a receiving node y_j) belongs to. Let $\mathcal{L}(N)$ (or $\mathcal{L}(x_i)$, $\mathcal{L}(y_j)$) denote the layer number where super node N (or x_i, y_j) belongs to. In a layered network, if $(x, y) \in \mathcal{E}$, $(x', y') \in \mathcal{E}$ and $\mathcal{L}(x) = \mathcal{L}(x')$, then we must have $\mathcal{L}(y) = \mathcal{L}(y') = \mathcal{L}(x) + 1 = \mathcal{L}(x') + 1$.

A cut, Ω, in a deterministic relay network $\mathcal{G}$ is a partition of the super nodes $\mathcal{N}$ (together with their incident nodes) into two disjoint sets Ω and Ω^c such that $S \in \Omega$ and $D \in \Omega^c$. For convenience, we call a cut a layer cut if all edges across the cut are emanating from nodes belonging to the same layer, otherwise we call it a cross-layer cut. We say an edge $(x_i, y_j) \in \mathcal{E}$ belongs to layer cut l if $\mathcal{L}(x_i) = l$. For a layered deterministic network, there are exactly $L - 1$ layer cuts.

The adjacency matrix T for a set of transmitting nodes $\mathbf{x} = \{x_1, x_2, ...x_m\}, x_i \in \mathcal{V}_x$, and a set of receiving nodes $\mathbf{y} = \{y_1, y_2, ...y_n\}, y_i \in \mathcal{V}_y$ in a deterministic relay network is a binary matrix of size $m \times n$ with rows corresponding to $\{x_i, x_i \in \mathbf{x}\}$ and columns corresponding to $\{y_i, y_i \in \mathbf{y}\}$ and $T(i, j) = 1$ if $(x_i, y_j) \in \mathcal{E}$. The adjacency matrix for a set of k edges is the binary adjacency matrix for the set of their transmitting nodes and the set of their receiving nodes.

A set of k edges are said to be linearly independent if the adjacency matrix for them has rank k, otherwise they are said to be linearly dependent. In a layered deterministic network, each $S - D$ path is of length $L - 1$ and crosses each layer cut exactly once. A set of k $S - D$ paths are said to be linearly independent if each set of their edges of size k crossing each layer cut are linearly independent, otherwise they are said to be linearly dependent. Lemma 4 shows that k linearly independent $S - D$ paths correspond to a transmission scheme of rate k.

In a deterministic wireless relay network, there are intermediate super nodes (exclude S and D) corresponding to the relay nodes in the original wireless relay network which have both transmitting nodes and receiving nodes. It is shown in Lemma 4 and Theorem 2 that one-bit length linear encoding functions at the relay super nodes are sufficient for constructing capacity-achieving transmission schemes between S and D for the underlying linear deterministic relay network.

Let $\mathcal{E}_\Omega$ be the set of edges crossing the cut Ω in a linear deterministic relay network. The cut value of Ω in the linear deterministic relay network is defined as the rank of the binary adjacency matrix for $\mathcal{E}_\Omega$, which equals the number of linearly independent edges in $\mathcal{E}_\Omega$. Note that here the cut value defined for linear deterministic wireless relay networks is different from that for graphs (which is just the number of edges crossing the cut). It is proved [1][2] that the unicast capacity of a linear deterministic wireless relay network is equal to the minimum cut value among all cuts.

III. Our Algorithm

A. Algorithm Outline

The max-flow min-cut theorem has been generalized to the linear deterministic wireless relay networks, but since the definitions of cut value for graphs (the number of edges crossing the cut) and for linear deterministic relay networks (the number of linearly independent edges crossing the cut) are different, the path augmentation algorithms for the latter are different from that for the former. As we will see later from this paper, the similarity is that in both cases the path augmentation algorithms operate in iterations and in each iteration they complete an additional path. The difference is that contrast to the simple addition of an available edge to a path in the path augmentation algorithms for graphs, the addition of edges to a path in a deterministic network has to satisfy some rank requirement of the adjacency matrix to avoid linear dependence among used path edges in each layer cut.

Our algorithm is basically a path augmentation algorithm for finding the maximum number K of linearly independent $S - D$ paths in a layered linear deterministic relay network, where K proves to be equal to C in Section IV-B. Moreover, the K identified paths correspond to a capacity-achieving transmission strategy for the underlying deterministic network.

The algorithm operates in iterations. In iteration k, a modified depth-first search (MDFS) algorithm tailored for the linear deterministic relay networks is carried out on the graph $\mathcal{G}$ trying to find an $S-D$ path in addition to the $k-1$ paths found in the first $k-1$ iterations (stored in a structure $\mathcal{P}$) so that the k found paths (stored in a structure $\mathcal{P}'$) are linearly independent. If MDFS returns True indicating that the kth $S-D$ path is found, then totally k linearly independent paths have been found and the algorithm continues to iteration $k+1$. If it returns False indicating that no $S-D$ path is found, then no more independent path exists and the algorithm stops while those identified paths in $\mathcal{P}'$ suggest a capacity-achieving transmission scheme.

B. Preliminaries

The following notations or structures are used in our algorithm. $\mathcal{P}$ and $\mathcal{P}'$ are structures storing information about the $k-1$ paths found in previous $k-1$ iterations and information about the updated $k-1$ paths and the partial path found in the current kth iteration respectively. Denote $\mathcal{E}_u$ ($\mathcal{E}_u^i$) as the set of edges in $\mathcal{E}$ used by paths in $\mathcal{P}'$ (in layer cut i). The MDFS algorithm ensures that edges in $\mathcal{E}_u^i$ are linearly independent, i.e., $\text{rank}(T(\mathcal{E}_u^i)) = |\mathcal{E}_u^i|$. Denote $\mathcal{V}_{xu}^i = \{x : (x,y) \in \mathcal{E}_u^i\}$ and $\mathcal{V}_{yu}^i = \{y : (x,y) \in \mathcal{E}_u^i\}$. $\mathcal{E}_u$ ($\mathcal{E}_u^i$), $\mathcal{V}_{xu}^i$, $\mathcal{V}_{yu}^i$ and $\mathcal{P}'$ are initialized at the beginning of every iteration according to $\mathcal{P}$ and maintained by MDFS in the current iteration subject to changes. As we will notice in Section III-C, each move MDFS makes ensures that $\text{rank}(T(\mathcal{E}_u^i)) = |\mathcal{E}_u^i|$. Let $\mathcal{E}_{u,x_j}^i$, $\mathcal{V}_{xu,x_j}^i$ and $\mathcal{V}_{yu,x_j}^i$ denote the instantaneous sets of $\mathcal{E}_u^i$, $\mathcal{V}_{xu}^i$ and $\mathcal{V}_{yu}^i$ when a transmitting node x_j is being explored in layer cut $i = \mathcal{L}(x_j)$ by MDFS in a certain iteration. Furthermore, denote $\mathcal{V}_{x,N} = \{x : x \in \mathcal{V}_x \text{ and } \mathcal{N}(x) = N\}$ and $\mathcal{V}_{y,N} = \{y : y \in \mathcal{V}_y \text{ and } \mathcal{N}(y) = N\}$. Denote $\mathcal{E}_{\mathcal{P}} = \{e : e \in \mathcal{E}, e \text{ used by some paths in } \mathcal{P}\}$ and $\mathcal{V}_{x\mathcal{P}} = \{x : (x,y) \in \mathcal{E}_{\mathcal{P}} \text{ for some } y\}$ and $\mathcal{V}_{y\mathcal{P}} = \{y : (x,y) \in \mathcal{E}_{\mathcal{P}} \text{ for some } x\}$.

Definition 1: $\mathcal{V}_{xspan}^{x_j}$: We define $\mathcal{V}_{xspan}^{x_j}$ as the set satisfies $\mathcal{V}_{xspan}^{x_j} \subseteq \mathcal{V}_{xu,x_j}^i$ and

$$T(x_j, \mathcal{V}_{yu,x_j}^i) = \sum_{x \in \mathcal{V}_{xspan}^{x_j}} T(x, \mathcal{V}_{yu,x_j}^i) \tag{1}$$

Since $T(\mathcal{V}_{xu,x_j}^i, \mathcal{V}_{yu,x_j}^i)$ has full rank, $\mathcal{V}_{xspan}^{x_j}$ is the unique such set.

Let function $\text{Span}(x_j)$ be the function for computing $\mathcal{V}_{xspan}^{x_j}$. In iteration $k+1$ of our algorithm, $|\mathcal{V}_{xu,x_j}^i| = k$.

Lemma 1: Let $|\mathcal{V}_{xu,x_j}^i| = k = |\mathcal{V}_{yu,x_j}^i|$. The computational complexity of $\text{Span}(x_j)$ for finding the set $\mathcal{V}_{xspan}^{x_j}$ is bounded by $O(k^3)$. For $\forall x \in \mathcal{V}_{xspan}^{x_j}$, $\text{rank}(T(\mathcal{V}_{xu,x_j}^i, \mathcal{V}_{yu,x_j}^i)) = \text{rank}(T(\mathcal{V}_{xu,x_j}^i +$

$x_j - x, \mathcal{V}^i_{yu,x_j})) = k$ and rank$(T(\mathcal{V}^{x_j}_{xspan} + x_j, \mathcal{V}^i_{yu,x_j})) =$ rank$(T(\mathcal{V}^{x_j}_{xspan}, \mathcal{V}^i_{yu,x_j})) =$ rank$(T(\mathcal{V}^{x_j}_{xspan} + x_j - x, \mathcal{V}^i_{yu,x_j})) = |\mathcal{V}^{x_j}_{xspan}| \leq k$.

Proof: To solve the set $\mathcal{V}^{x_j}_{xspan}$ is equivalent to solving the system of linear equations, $\mathcal{V}^{x_j}_{xspan} \cdot T(\mathcal{V}^i_{xu,x_j}, \mathcal{V}^i_{yu,x_j}) = T(x_j, \mathcal{V}^i_{yu,x_j})$ in $GF(2)$ which can be accomplished in time $O(k^3)$ by using Gaussian elimination. The second statement is obvious. ■

Consider the subgraph consisting of nodes $x_j \cup \mathcal{V}^i_{xu,x_j} \cup \mathcal{V}^i_{yu,x_j}$ and the edges connecting them in $\mathcal{G}$. Let $\mathcal{G}^{x_j}_{sub}$ denote the graph obtained by reversing the directions of the edges in $\mathcal{E}^i_{xu,x_j}$ in the above subgraph.

Lemma 2: There is a directed path from x_j to any $x \in \mathcal{V}^{x_j}_{xspan}$ in graph $\mathcal{G}^{x_j}_{sub}$. Let FindIndPaths be the function for finding out all these $|\mathcal{V}^{x_j}_{xspan}|$ paths from x_j. The computational complexity of FindIndPaths$(x_j, \mathcal{V}^{x_j}_{xspan})$ is bounded by $O(k^2)$ in iteration k.

Proof: From Lemma 1, for $\forall x \in \mathcal{V}^{x_j}_{xspan}$, rank$(T(\mathcal{V}^i_{xu,x_j}, \mathcal{V}^i_{yu,x_j})) =$ rank$(T(\mathcal{V}^i_{xu,x_j} + x_j - x, \mathcal{V}^i_{yu,x_j})) = k$ where $k = |\mathcal{P}|$. Introduce an auxiliary receiving node y' and an edge (x, y'). It's easy to see that rank$(T(\mathcal{V}^i_{xu,x_j} + x_j, \mathcal{V}^i_{yu,x_j} + y')) = k + 1$. Given rank$(T(\mathcal{V}^i_{xu,x_j}, \mathcal{V}^i_{yu,x_j})) = k$ and rank$(T(\mathcal{V}^i_{xu,x_j} + x_j, \mathcal{V}^i_{yu,x_j} + y')) = k + 1$, there is a size k prefect matching between $\mathcal{V}^i_{xu,x_j}$ and $\mathcal{V}^i_{yu,x_j}$, $M_1 = \mathcal{E}^i_{u,x_j}$ being such a matching, and a size $k + 1$ perfect matching between $\mathcal{V}^i_{xu,x_j} + x_j$ and $\mathcal{V}^i_{yu,x_j} + y'$. Using a similar argument as in finding the maximum bipartite matching, we claim that there is an alternating path, relative to the matching M_1, starting from an unused transmitting node x_j to an unused receiving node y', alternating between edges not in the current matching M_1 and edges in the current matching M_1, i.e., there is a path $P_{x_j \to y'} = \{(x_j, y_1), (y_1, x_1), (x_1, y_2), (y_2, x_2), ...(x_{m-1}, y_m), (y_m, x_m), (x_m, y') = (x, y')\}$ with $(x_i, y_i), 1 \leq i \leq m$ being edges in $M_1 = \mathcal{E}^i_{u,x_j}$. So we proved that there is a path $P_{x_j \to x} = \{(x_j, y_1), (y_1, x_1), (x_1, y_2), (y_2, x_2), ...(x_{m-1}, y_m), (y_m, x_m) = (y_m, x)\}$ with $(x_i, y_i), 1 \leq i \leq m$ being edges in $\mathcal{E}^i_{u,x_j}$. Without loss of generality, we also use the following representation of the path $P_{x_j \to x} = \{(x_j, y_1), (x_1, y_1), (x_1, y_2), (x_2, y_2), ...(x_{m-1}, y_m), (x_m, y_m) = (x, y_m)\}$ with $(x_i, y_i), 1 \leq i \leq m$ being edges in $\mathcal{E}^i_{u,x_j}$.

In iteration k of our algorithm, $|\mathcal{V}^i_{xu}| = |\mathcal{V}^i_{yu}| = k - 1$, so the number of nodes in $\mathcal{G}^{x_j}_{sub}$ is bounded by $O(k)$, which also means the number of edges in $\mathcal{G}^{x_j}_{sub}$ is bounded by $O(k^2)$. To find directed paths from x_j to all $x \in \mathcal{V}^{x_j}_{xspan}$ in $\mathcal{G}^{x_j}_{sub}$ takes time bounded by $O(k^2)$ by using some well-known graph traversal algorithms, like breadth-first search. ■

Let's briefly recall the depth-first search (DFS) algorithm first. DFS algorithm is a well-known algorithm for traversing graphs. To find a path, DFS progresses by exploring the outgoing edges of a node before exploring any other outgoing edges of the node's predecessor. In this way, it proceeds deeper and deeper until it reaches the goal node when it stops or until it reaches a node without any outgoing edges when it backtracks to the most recent node that it hasn't finished exploring. In the search process, each node in the graph would be explored at most once.

Our modified depth-first search (MDFS) algorithm for linear deterministic relay networks inherits the basic forwarding and backtracking operations from the basic DFS algorithm and each super node in $\mathcal{N}$ is treated like a node in DFS. The difference is that in order to find an $S-D$ path in the graph $\mathcal{G}=(\mathcal{N},\mathcal{V},\mathcal{E})$ linearly independent to the paths in $\mathcal{P}$, we have to avoid linear dependency between different paths in our algorithm, which means that instead of allowing a valid forwarding move along each outgoing edge when a node is explored as in DFS, more constraints should be imposed on the forwarding moves of a super node in MDFS. We propose the following MDFS algorithm to accomplish the path augmentation task in our problem.

C. Modified Depth-First Search (MDFS) Algorithm

In the following, the exploration to a super node $N\in\mathcal{N}$ or a node $v\in\mathcal{V}$ by MDFS refers to that MDFS has extended the path found in the current iteration to N or $\mathcal{N}(v)$ and now it continues to extend the path further from N or from v. The exploration to a super node N is realized as the exploration to some of its unexplored incident nodes v (which we call admissible nodes of N in Definition 2). Once a super node or a node is explored, it's labeled explored. During the running time of MDFS, each super node $N\in\mathcal{N}$ could be labeled as unexplored or explored indicating that it hasn't or has been explored by MDFS. Each node in $\mathcal{V}$ could be labeled as unexplored or explored indicating that it doesn't allow or allows exploration by MDFS. There is also a type labeling with each node in $\mathcal{V}_x$ indicating how it can be explored by MDFS if it allows exploration. Let SetLabel(X, LABEL) be the function setting label of X (X can be a super node or a node) as LABEL (LABEL can be explored or unexplored). Let LABEL=GetLabel(X) be the function returning the label of X. Let SetType(x,B) be the function setting type of transmitting node x as B (B can be $1,2,$ or 3). Let $B=$GetType(x) be the function returning the type of x.

Definition 2: **Admissible nodes and admissible forwarding moves of super nodes –** We define the admissible nodes for a super node N when N is explored by MDFS as follows: $\mathcal{V}_N^{ad} = \{x : x \in \mathcal{V}_{x,N}, x \notin \mathcal{V}_{xu}^{\mathcal{L}(N)}, x$ labeled unexplored or $y : y \in \mathcal{V}_{y,N}, y \in \mathcal{V}_{y\mathcal{P}}, y$ labeled unexplored$\}$. The exploration to a super node N is realized as the exploration to its admissible nodes. The following forwarding moves starting from $\mathcal{V}_N^{ad}$ are defined as admissible forwarding moves allowed in MDFS algorithm on the graph $\mathcal{G} = (\mathcal{N}, \mathcal{V}, \mathcal{E})$ when it explores a super node N.

1) *Type* 1: moving forward along $(x, y) \in \mathcal{E}$ with $x \in \mathcal{V}_N^{ad}$, GetType$(x) = 1, 2,$ or 3 and $y \notin \mathcal{V}_{yu}^{\mathcal{L}(N)}$, rank$(T(\mathcal{V}_{xu,x}^{\mathcal{L}(N)} + x, \mathcal{V}_{yu,x}^{\mathcal{L}(N)} + y)) = |\mathcal{P}| + 1$ and GetLabel$(\mathcal{N}(y))$ =unexplored.
2) *Type* 2: moving forward along the path $P_{x \to x'}$ (as proved to exist in Lemma 2) for any $x' \in \mathcal{V}_{xspan}^{x}$ with $x \in \mathcal{V}_N^{ad}$, GetType$(x) = 1$ or 3.
3) *Type* 3: moving forward along $(x, y) \in \mathcal{E}$ with $y \in \mathcal{V}_N^{ad}$, $(x, y) \in \mathcal{E}_{u,x}^{\mathcal{L}(N)-1}$.

When N is explored, $x \in \mathcal{V}_N^{ad}$ with GetType$(x) = 1$ or 3 would allow to start type 1 and type 2 admissible forwarding moves, $x \in \mathcal{V}_N^{ad}$ with GetType$(x) = 2$ would allow to start type 1 admissible forwarding moves only and $y \in \mathcal{V}_N^{ad}$ would allow to start type 3 admissible forward moves. There is an ordering in exploring $\mathcal{V}_N^{ad}$: type 2 and type 3 nodes should be explored first, then type 1 nodes and finally the receiving nodes.

Definition 3: **Modified depth-first search algorithm (MDFS):** The MDFS algorithm is defined in terms of initialization, exploring of a super node, labeling of $\mathcal{N}, \mathcal{V}$ and updating of $\mathcal{P}'$ and $\mathcal{E}_u$ as follows:

1) *Initialization:* Set $\mathcal{N}$ and $\mathcal{V}$ as unexplored and $\mathcal{V}_x$ as type 1 nodes.
2) *Exploring of a super node N:* When MDFS explores a super node N, it means that a partial path $P_{|\mathcal{P}|+1}$ from S to N has been found in addition to the $|\mathcal{P}|$ complete $S - D$ paths. As mentioned in Definition 2, the exploration to a super node N is realized as exploration to its admissible nodes $\mathcal{V}_N^{ad}$ and three kinds of admissible forwarding moves from $\mathcal{V}_N^{ad}$ are allowed. There is an ordering in exploring $\mathcal{V}_N^{ad}$: type 2 and type 3 nodes should be explored first, then type 1 nodes and finally the receiving nodes. Once a super node or a node is explored by MDFS, it is labeled explored. Now we explain how to understand these three admissible forwarding moves and how to label $\mathcal{N}, \mathcal{V}$ and update $\mathcal{P}'$, $\mathcal{E}_u$ with these moves.

a) Type 1 admissible forwarding move: A type 1 admissible forwarding move can be understood as that MDFS extends $P_{|\mathcal{P}|+1}$ along an edge (x, y) from super node N to super node $\mathcal{N}(y)$, i.e., $P_{|\mathcal{P}|+1} = P_{|\mathcal{P}|+1} + (x, y)$. $\mathcal{P}'$ is updated accordingly. Then MDFS makes $\mathcal{N}(y)$ with $\mathcal{L}(\mathcal{N}(y)) = \mathcal{L}(N) + 1$ the next super node to be explored when $\mathcal{E}_u^{\mathcal{L}(N)}$ is updated as $\mathcal{E}_u^{\mathcal{L}(N)} + (x, y)$ and $|\mathcal{E}_u^{\mathcal{L}(N)}|$ increases from $|\mathcal{P}|$ to $|\mathcal{P}| + 1$.

b) Type 2 admissible forwarding move: A type 2 admissible forwarding move can be understood as that MDFS updates $\mathcal{P}'$ according to the path $P_{x\to x'}$ (proved to exist in Lemma 2) for some $x' \in \mathcal{V}_{xspan}^x$ as follows. Let $P_{x\to x'} = \{(x, y_1), (x_1, y_1), (x_1, y_2), ...(x_m, y_m) = (x', y_m)\}$ with $(x_i, y_i), 1 \le i \le m$ being path edges used by m $S - D$ paths in $\mathcal{P}'$ existing before the current move, denoted as $P_i, 1 \le i \le m$. After the current type 2 forwarding move, $P_i, 1 \le i \le m$ and $P_{|\mathcal{P}|+1}$ are updated to P_i' and $P'_{|\mathcal{P}|+1}$ as follows. Let $P_i(N_1, N_2)$ denote the segment of P_i from super node N_1 to N_2. $P_1' = P_{|\mathcal{P}|+1}(S, \mathcal{N}(x)) + (x, y_1) + P_1(\mathcal{N}(y_1), D)$, $P_i' = P_{i-1}(S, \mathcal{N}(x_{i-1})) + (x_{i-1}, y_i) + P_i(\mathcal{N}(y_i), D), 1 < i \le m$ and $P'_{|\mathcal{P}|+1} = P_m(S, \mathcal{N}(x'))$. $\mathcal{P}'$ is updated accordingly. After the type 2 admissible forwarding move, x' is labeled as unexplored and set as type 2 node. Then MDFS makes $\mathcal{N}(x')$ with $\mathcal{L}(\mathcal{N}(x')) = \mathcal{L}(N)$ the next super node to be explored when $\mathcal{E}_u^{\mathcal{L}(N)}$ is updated as $\mathcal{E}_u^{\mathcal{L}(N)} + (x, y_1) - (x_1, y_1) + (x_1, y_2) - (x_2, y_2) + ... + (x_{m-1}, y_m) - (x', y_m)$ and $|\mathcal{E}_u^{\mathcal{L}(N)}|$ keeps to be $|\mathcal{P}|$.

c) Type 3 admissible forwarding move: A type 3 admissible forwarding move can be understood as that MDFS updates $\mathcal{P}'$ along the edge (x, y) as follows. Let P_x be the path in $\mathcal{P}'$ existing before the current move where (x, y) belongs. If $P_x \neq P_{|\mathcal{P}|+1}$, then after the current move, MDFS updates P_x and $P_{|\mathcal{P}|+1}$ to P_x' and $P'_{|\mathcal{P}|+1}$ as follows. $P_x' = P_{|\mathcal{P}|+1}(S, N) + P_x(N, D)$ and $P'_{|\mathcal{P}|+1} = P_x(S, \mathcal{N}(x))$. If $P_x = P_{|\mathcal{P}|+1}$, then after the current move, MDFS updates $P_{|\mathcal{P}|+1}$ to $P'_{|\mathcal{P}|+1}$ as follows. $P'_{|\mathcal{P}|+1} = P_{|\mathcal{P}|+1}(S, \mathcal{N}(x))$. $\mathcal{P}'$ is updated accordingly. After the type 3 move, y is labeled explored, x is labeled as unexplored and set as type 3 node. Then MDFS makes $\mathcal{N}(x)$ with $\mathcal{L}(\mathcal{N}(x)) = \mathcal{L}(N) - 1$ the next super node to be explored when $\mathcal{E}_u^{\mathcal{L}(N)-1}$ is updated as $\mathcal{E}_u^{\mathcal{L}(N)-1} - (x, y)$ and $|\mathcal{E}_u^{\mathcal{L}(N)-1}|$ decreases from $|\mathcal{P}| + 1$ to $|\mathcal{P}|$.

Lemma 3: In iteration $k + 1$ of our algorithm, MDFS defined in Definition 3 maintains a set of size k linearly independent $S - D$ paths while it tries to complete a $(k + 1)$th $S - D$ path P_{k+1}. And when MDFS explores a super node N, it means MDFS extends P_{k+1} from S to N

in addition to k complete $S-D$ paths (all stored in $\mathcal{P}'$) and there are $k+1$ linearly independent used path edges in each of the first $\mathcal{L}(N)-1$ layer cuts and k linearly independent used path edges in each of the rest layer cuts. These two facts lead to the conclusion that when MDFS returns True in iteration $k+1$, totally $k+1$ linearly independent $S-D$ paths are found (and stored in $\mathcal{P}'$).

Proof: We prove by induction. Assume that in the first k iterations of our algorithm, MDFS already finds k linearly independent $S-D$ paths. Then at the beginning of iteration $k+1$ before we call MDFS, these two statements are clearly true. Now it's sufficient for us to show that these three kinds of forwarding moves allowed in MDFS keep these two statements true after each forwarding move of MDFS. First, clearly a type 1 forwarding move along an edge (x,y) increases $|\mathcal{E}_u^{\mathcal{L}(x)}|$ from k to $k+1$ with rank$(T(\mathcal{E}_u^{\mathcal{L}(x)}))=k+1$ and it has no impact to the set of k $S-D$ paths existing before the current move. So these two statements remain true after a type 1 forwarding move. Second, a type 2 forwarding move keeps $|\mathcal{E}_u^{\mathcal{L}(x)}|$ unchanged with rank$(T(\mathcal{E}_u^{\mathcal{L}(x)}))=k$ (according to Lemma 1) while it updates the paths $P_i, 1\le i\le m$ and P_{k+1} to $P'_i, 1\le i\le m$ and P'_{k+1}. Clearly the rest $k-m$ $S-D$ paths existing before the current move are not impacted by the move. So these two statements remain true after a type 2 forwarding move. Third, a type 3 forwarding move extends P_{k+1} along an edge (x,y) from super node $\mathcal{N}(y)$ to $\mathcal{N}(x)$. Because MDFS reaches $\mathcal{N}(y)$ before the current move, $|\mathcal{E}_u^{\mathcal{L}(\mathcal{N}(x)}|=k+1$ and after the current move, $|\mathcal{E}_u^{\mathcal{L}(\mathcal{N}(x)}|=k$ with (x,y) deleted. Clearly MDFS maintains k linearly independent $S-D$ paths after the current move when it proceeds to explore $\mathcal{N}(x)$. So these two statements remain true after a type 3 forwarding move. ■

Lemma 4: Let $\mathcal{P}$ be the set of $S-D$ paths found by our algorithm and $|\mathcal{P}|=K$, then the paths in $\mathcal{P}$ correspond to some transmission scheme of rate K from S to D for the underlying deterministic network.

Proof: We prove by constructing a transmission scheme of rate K from S to D for the underlying deterministic network by using the K paths in $\mathcal{P}$. Lemma 3 says that these K paths in $\mathcal{P}$ found by our algorithm are linearly independent. Let $\mathcal{V}_{y,N}^{\mathcal{P}}\subseteq\mathcal{V}_{y,N}$ ($\mathcal{V}_{x,N}^{\mathcal{P}}\subseteq\mathcal{V}_{x,N}$) be the set of transmitting (receiving) nodes incident to N that are used by some paths in $\mathcal{P}$. Clearly $|\mathcal{V}_{y,N}^{\mathcal{P}}|=|\mathcal{V}_{x,N}^{\mathcal{P}}|$. The relay function at each relay super node N could be any one-one mapping between $\mathcal{V}_{y,N}^{\mathcal{P}}$ and $\mathcal{V}_{x,N}^{\mathcal{P}}$, i.e., each received bit from $\mathcal{V}_{y,N}^{\mathcal{P}}$ is transmitted forward by a unique transmitting node in $\mathcal{V}_{x,N}^{\mathcal{P}}$ specified by the mapping. Clearly each such mapping corresponds to

a full rank adjacency matrix. For simplicity, we can treat those relay functions as intra-layer paths which together with the paths in $\mathcal{P}$ (treated as inter-layer paths) completely specify a transmission scheme from S to D. Since in each of the $L-1$ inter-layers and each of the $L-2$ intra-layers the adjacency matrix for used path edges has full rank K, the overall transfer matrix from S to D along these paths is the product of all these adjacency matrix and also has full rank K which guarantees that transmission information rate K is allowed between $S-D$ with the transmission scheme defined by these K paths in $\mathcal{P}$ and any one-one mapping function for each relay super node. ■

Let's justify the rule that type 2 nodes shouldn't start any type 2 forwarding moves in MDFS. By definition, a type 1 or type 3 transmitting node $x \in \mathcal{V}_N^{ad}$ can start type 1 or type 2 admissible forwarding moves of N while a type 2 transmitting node $x' \in \mathcal{V}_N^{ad}$ can only start type 1 admissible forward moves of N. Lemma 1 helps in explaining why type 2 transmitting node shouldn't start any type 2 forwarding moves. When x' is labeled as type 2 node after a type 2 forwarding move along the path $P_{x\to x'}$ for some type 1 or type 3 node x, it means that $x' \in \mathcal{V}_{xspan}^{x}$. It's easy to see that when x' is explored right after the type 2 forwarding move along path $P_{x\to x'}$, $\mathcal{V}_{xspan}^{x'} = \mathcal{V}_{xspan}^{x} + x - x'$. Since x is explored and x is of type 1 or type 3, all transmitting nodes in $\mathcal{V}_{xspan}^{x}$ could be explored following type 2 forwarding moves from x. It means that all transmitting nodes in $\mathcal{V}_{xspan}^{x'}$ are guaranteed to have the chance of being explored by MDFS as long as x is explored and all type 2 forwarding moves from x are allowed. So type 2 node x' shouldn't start any type 2 forwarding moves to avoid redundancy.

D. Implementation Details of MDFS

The MDFS algorithm is implemented in the (Res,$\mathcal{P}'$) =MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', N, D, k$) function. The input parameters are the graph $\mathcal{G}$, information about the $k-1$ found $S-D$ paths stored in structure $\mathcal{P}$, information about the updated $k-1$ $S-D$ paths and partial path P_k stored in a structure $\mathcal{P}'$, starting super node N from where to complete P_k, ending super node D where to end P_k and the iteration number k. In each iteration, we call MDFS by initializing N to be S for finding an $S-D$ path. The function returns (True, $\mathcal{P}'$) with $|\mathcal{P}'| = |\mathcal{P}| + 1 = k$ if the kth $S-D$ path is found in current iteration and (False, $\mathcal{P}'$) with $\mathcal{P}' = \mathcal{P}$ if no path is found. Our algorithm calls MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', S, D, k$) in iteration k for increasing k until MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', S, D, K+1$) returns Res=False for some K, then our algorithm stops and claims that $K = C$. Each time

MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', S, D, k$) returns Res=True, $\mathcal{P}$ is updated as $\mathcal{P}'$ and used in next iteration.

E. Accelerating The MDFS Algorithm

From definition, we see that the main computational complexity of the MDFS algorithm comes from the rank computation of the associated binary adjacency matrix in deciding whether an edge (x, y) qualifies for a type 1 admissible forwarding move when x of type 1, type 2 or type 3 is explored and from the computation of $\mathcal{V}^x_{xspan}$ in deciding whether a type 2 admissible forwarding move exists between a pair x, x' when x of type 1 or type 3 is explored by MDFS. In this subsection we explore some useful combinatorial properties related to the rank of the binary adjacency matrix and develop an equivalent but computationally simple method to replace the rank computation in MDFS.

In iteration $k+1$ of our algorithm, let $Y^{k+1}_{x_j} = \{y : \mathcal{L}(y) = \mathcal{L}(x_j)+1 = i+1$ and $y \notin \mathcal{V}^i_{yu,x_j}\}$, let $Y^{k+1}_{x_j,c} = \{y : y \in Y^{k+1}_{x_j}$ and $(x, y) \in \mathcal{E}$ for some $x \in \{x_j \cup \mathcal{V}^{x_j}_{xspan}\}\}$ and let $Y^{k+1}_{x_j,fr} = \{y : y \in Y^{k+1}_{x_j}$ and rank$(T(\mathcal{V}^i_{xu,x_j} + x_j, \mathcal{V}^i_{yu,x_j} + y)) = k + 1\}$.

Lemma 5: $Y^{k+1}_{x_j,fr}$ is a subset of $Y^{k+1}_{x_j,c}$. The rank check of the adjacency matrix (rank$(T(\mathcal{V}^i_{xu,x_j} + x_j, \mathcal{V}^i_{yu,x_j}+y)) = k$ or k+1) for $\forall y \in Y^{k+1}_{x_j,c}$ is equivalent to checking $T(x_j, y) = \sum_{x\in\mathcal{V}^{x_j}_{xspan}} T(x, y)$ or not, whose computational complexity is bounded by $O(k)$.

Proof: First we prove that for $\forall y \in Y^{k+1}_{x_j}$, if there is no $(x, y) \in \mathcal{E}$ with $x \in \{x_j \cup \mathcal{V}^{x_j}_{xspan}\}$, then $T(\mathcal{V}^i_{xu,x_j} + x_j, \mathcal{V}^i_{yu,x_j} + y) = k$. If there is no $(x, y) \in \mathcal{E}$ with $x \in \{x_j \cup \mathcal{V}^{x_j}_{xspan}\}$, then the vector $T(\mathcal{V}^i_{xu,x_j} + x_j, y)$ has zero entries on rows indexed by $x_j \cup \mathcal{V}^{x_j}_{xspan}$, together with equation (1) we have $T(x_j, \mathcal{V}^i_{yu,x_j}+y) = \sum_{x\in\mathcal{V}^{x_j}_{xspan}} T(x, \mathcal{V}^i_{yu,x_j}+y)$. So rank$(T(\mathcal{V}^i_{xu,x_j}+x_j, \mathcal{V}^i_{yu,x_j}+y)) =$ rank$(T(\mathcal{V}^i_{xu,x_j}, \mathcal{V}^i_{yu,x_j} + y)) = k$. So we proved $Y^{k+1}_{x_j,fr}$ is a subset of $Y^{k+1}_{x_j,c}$.

We know rank$(T(\mathcal{V}^i_{xu,x_j}, \mathcal{V}^i_{yu,x_j}+y)) = k$ and $|\mathcal{V}^i_{xu,x_j}| = k$. If rank$(T(\mathcal{V}^i_{xu,x_j}+x_j, \mathcal{V}^i_{yu,x_j}+y)) = k$, then it must be $T(x_j, \mathcal{V}^i_{yu,x_j} + y) = \sum_{x\in\mathcal{V}^{x'_j}_{xspan}} T(x, \mathcal{V}^i_{yu,x_j} + y)$ for some $\mathcal{V}^{x'_j}_{xspan} \subseteq \mathcal{V}^i_{xu,x_j}$. Given equation (1) and $\mathcal{V}^{x_j}_{xspan}$ is unique, we have $\mathcal{V}^{x'_j}_{xspan} = \mathcal{V}^{x_j}_{xspan}$ and $T(x_j, \mathcal{V}^i_{yu,x_j} + y) = \sum_{x\in\mathcal{V}^{x'_j}_{xspan}} T(x, \mathcal{V}^i_{yu,x_j} + y)$ is equivalent to $T(x_j, y) = \sum_{x\in\mathcal{V}^{x_j}_{xspan}} T(x, y)$. So rank$(T(\mathcal{V}^i_{xu,x_j} + x_j, \mathcal{V}^i_{yu,x_j} + y)) = k$ or $k + 1$ is equivalent to $T(x_j, y) = \sum_{x\in\mathcal{V}^{x_j}_{xspan}} T(x, y)$ or not. ■

Lemma 5 says that given $\mathcal{V}^x_{xspan}$ we can simplify the rank computation of the associated binary adjacency matrix in deciding whether an edge (x, y) qualifies for a type 1 admissible forwarding move when x of type 1, type 2 or type 3 is explored. We also know that when x' is explored as a type 2 node right after the type 2 forwarding move along path $P_{x\to x'}$ for some x of type 1

or type 3, $\mathcal{V}_{xspan}^{x'} = \mathcal{V}_{xspan}^{x} + x - x'$, so we don't need to compute $\mathcal{V}_{xspan}^{x'}$ for any x' of type 2 being explored as long as $\mathcal{V}_{xspan}^{x}$ is computed.

Lemma 6: In the last iteration $K+1$ of our algorithm when no more $S-D$ path is found by MDFS, all $\mathcal{N}(x_k)$ with $x_k \in \mathcal{V}_{xspan}^{x_j}$ and all $\mathcal{N}(y)$ with $y \in Y_{x_j,fr}^{K+1}$ must have been explored by MDFS before MDFS returns if x_j has ever been explored.

Proof: Since there is no more $S-D$ path exists in the last iteration of our algorithm, all possible admissible moves would be tried by MDFS. It means each $\mathcal{N}(x_k)$ with $x_k \in \mathcal{V}_{xspan}^{x_j}$ would be explored by following a type 2 forwarding move along the path $P_{x_j \to x_k}$ if x_j of type 1 or type 3 has ever been explored. For any x_i of type 2 being explored right after a type 2 move along the path $P_{x_j \to x_i}$ starting from some x_j of type 1 or type 3, we know that x_j is explored and $\mathcal{V}_{xspan}^{x_i} = \mathcal{V}_{xspan}^{x_j} + x_j - x_i$. So if all $\mathcal{N}(x_k)$ with $x_k \in \mathcal{V}_{xspan}^{x_j}$ are explored, all $\mathcal{N}(x'_k)$ with $x'_k \in \mathcal{V}_{xspan}^{x_i}$ are also explored.

From Lemma 5, for $\forall y \in Y_{x_j,fr}^{K+1}$, there is $(x,y) \in \mathcal{E}$ for some $x \in \{x_j \cup \mathcal{V}_{xspan}^{x_j}\}$. We already proved that all $\mathcal{N}(x_k)$ with $x_k \in \mathcal{V}_{xspan}^{x_j}$ would be explored by MDFS in the last iteration of our algorithm if x_j has ever been explored. So the edge (x,y) with $x \in \{x_j \cup \mathcal{V}_{xspan}^{x_j}\}$ and $y \in Y_{x_j,fr}^{K+1}$ must be considered by MDFS in the last iteration given that x_j has ever been explored. If $x = x_j$, (x,y) allows a type 1 forwarding move when x_j is explored and then $\mathcal{N}(y)$ will be explored. Now assume $x \in \mathcal{V}_{xspan}^{x_j}$. Given x_j is explored, x is also explored and $\mathcal{V}_{xu,x}^{i} + x = \mathcal{V}_{xu,x_j}^{i} + x_j$ and $\mathcal{V}_{yu,x}^{i} = \mathcal{V}_{yu,x_j}^{i}$. Given $y \in Y_{x_j,fr}^{K+1}$, i.e., rank$(T(\mathcal{V}_{xu,x_j}^{i} + x_j, \mathcal{V}_{yu,x_j}^{i} + y)) = K+1$, we have rank$(T(\mathcal{V}_{xu,x}^{i} + x, \mathcal{V}_{yu,x}^{i} + y)) = K+1$. Then (x,y) allows a type 1 forwarding move when x is explored and then $\mathcal{N}(y)$ will be explored. So all $\mathcal{N}(y)$ with $y \in Y_{x_j,fr}^{K+1}$ must be explored by MDFS in the last iteration of our algorithm if x_j has ever been explored. ■

Table I gives a pseudo-code description of the MDFS algorithm described above. Refer to [5] for an implementation of the algorithm proposed in the current paper.

IV. Algorithm Analysis

A. Complexity Analysis

Theorem 1: MDFS algorithm defined in Section III terminates in finite time as the total number of explorations to transmitting nodes invoked by MDFS is bounded by $O(|\mathcal{V}_x| \cdot k)$ in iteration k of our algorithm. The total computational complexity of our algorithm is bounded

TABLE I

(Res={True, False},$\mathcal{P}'$)=**MDFS**($\mathcal{G}, \mathcal{P}, \mathcal{P}', N, D, k$)

SetLabel(N, explored)

for $\forall x \in \mathcal{V}_{x,N}$ with $x \notin \mathcal{E}_{xu}^{\mathcal{L}(N)}$ and GetLabel(x) = unexplored and GetType(x) = 2

- SetLabel(x, explored)
- for $\forall e = (x, y) \in \mathcal{G}$.incidentEdgeType1($x$) and GetLabel($\mathcal{N}(y)$) = unexplored
  - if $T(x, y) \neq \sum_{x' \in \mathcal{V}_{xspan}^{x}} T(x', y)$
    - $\mathcal{E}_u^{\mathcal{L}(N)} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)} + e$; Update($\mathcal{P}'$)
    - if $\mathcal{N}(y) = D$ { return (True,$\mathcal{P}'$) }
    - else
      - (Res, $\mathcal{P}'$) = MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', \mathcal{N}(y), D, k$)
      - if Res = True { return (True,$\mathcal{P}'$) }
    - $\mathcal{E}_u^{\mathcal{L}(N)} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)} - e$; Restore($\mathcal{P}'$)

for $\forall x \in \mathcal{V}_{x,N}$ with $x \notin \mathcal{E}_{xu}^{\mathcal{L}(N)}$ and GetLabel(x) = unexplored and GetType(x) $\neq 2$

(first for x with GetType(x) = 3, then for x with GetType(x) = 1)

- SetLabel(x, explored)
- $\mathcal{V}_{xspan}^{x}$ = Span(x)
- for $\forall e = (x, y) \in \mathcal{G}$.incidentEdgeType1($x$) and GetLabel($\mathcal{N}(y)$) = unexplored
  - if $T(x, y) \neq \sum_{x' \in \mathcal{V}_{xspan}^{x}} T(x', y)$
    - $\mathcal{E}_u^{\mathcal{L}(N)} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)} + e$; Update($\mathcal{P}'$)
    - if $\mathcal{N}(y) = D$ { return (True,$\mathcal{P}'$) }
    - else
      - (Res, $\mathcal{P}'$) = MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', \mathcal{N}(y), D, k$)
      - if Res = True { return (True,$\mathcal{P}'$) }
    - $\mathcal{E}_u^{\mathcal{L}(N)} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)} - e$; Restore($\mathcal{P}'$)
- $P_{x \to \mathcal{V}_{xspan}^{x}}$ = FindIndPaths($x, \mathcal{V}_{xspan}^{x}$)
- for $\forall x' \in \mathcal{V}_{xspan}^{x}$ with $P_{x \to x'} = \{e_1, e_2, ...e_{2m}\} = \{(x, y_1), (y_1, x_1), (x_1, y_2), ...(y_m, x_m) = (y_m, x')\}$
  - SetLabel(x', unexplored); SetType(x', 2); $\mathcal{V}_{xspan}^{x'} = \mathcal{V}_{xspan}^{x} - x' + x$
  - $\mathcal{E}_u^{\mathcal{L}(N)} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)} + e_1 - e_2 + e_3 - e_4 + ... + e_{2m-1} - e_{2m}$; Update($\mathcal{P}'$)
  - (Res, $\mathcal{P}'$) = MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', \mathcal{N}(x'), D, k$)
  - if Res = True { return (True,$\mathcal{P}'$) }
  - $\mathcal{E}_u^{\mathcal{L}(N)} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)} - e_1 + e_2 - e_3 + e_4 - ... - e_{2m-1} + e_{2m}$; Restore($\mathcal{P}'$)

for $\forall e = (x, y) \in \mathcal{G}$.incidentEdgeType3($N$) with GetLabel($y$) = unexplored

- SetLabel(y, explored)
- SetLabel(x, unexplored); SetType(x, 3)
- $\mathcal{E}_u^{\mathcal{L}(N)-1} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)-1} - e$; Update($\mathcal{P}'$)
- (Res, $\mathcal{P}'$) = MDFS($\mathcal{G}, \mathcal{P}, \mathcal{P}', \mathcal{N}(x), D, k$)
- if Res = True { return (True,$\mathcal{P}'$) }
- $\mathcal{E}_u^{\mathcal{L}(N)-1} \leftarrow \mathcal{E}_u^{\mathcal{L}(N)-1} + e$; Restore($\mathcal{P}'$)

return (False,$\mathcal{P}'$)

by $O(|\mathcal{V}_x| \cdot C^4 + d \cdot |\mathcal{V}_x| \cdot C^3)$ if our algorithm stops after finding C linearly independent $S - D$ paths.

Proof: In MDFS, a transmitting node is explored or re-explored exclusively as an unexplored type 1, type 2 or type 3 node. Let k_1, k_2 and k_3 be the total number of labeling of a transmitting node as unexplored type 1, type 2 and type 3 node respectively. To prove MDFS algorithm terminates, we can equivalently prove that k_1, k_2 and k_3 are finite.

Claim 1: $k_1 \leq |\mathcal{V}_x|$.

Proof: All nodes in $\mathcal{V}_x$ are labeled as unexplored type 1 nodes at the beginning of the iteration. After an unexplored type 1 node is explored, it's labeled as explored and never relabeled as unexplored type 1 node again. So $k_1 \leq |\mathcal{V}_x|$. ■

Claim 2: $k_3 \leq |\mathcal{V}_x|$.

Proof: A node is labeled as unexplored type 3 node only after a type 3 admissible forwarding move. We next show that the number of type 3 admissible forwarding moves is bounded by $|\mathcal{V}_x|$. A type 3 admissible forwarding move only happens on an edge (x, y) with $y \in \mathcal{V}_{y\mathcal{P}}$ unexplored and y is labeled as explored after a type 3 move and never relabeled as unexplored again. Since $|\mathcal{V}_{y\mathcal{P}}| = |\mathcal{V}_{x\mathcal{P}}| \leq |\mathcal{V}_x|$, so the total number of type 3 moves is bounded by $|\mathcal{V}_x|$. The total number of labeling of transmitting nodes as unexplored type 3 nodes is therefore bounded by $|\mathcal{V}_x|$. ■

Claim 3: $k_2 \leq k \cdot (k_1 + k_3)$.

Proof: A node is labeled as unexplored type 2 node only after a type 2 admissible forwarding move. We next show that the total number of type 2 moves is bounded by $k_2 \leq k \cdot (k_1 + k_3)$. A type 2 admissible forwarding move only starts from some type 1 or type 3 node being explored. The total number of type 1 or type 3 nodes explored by MDFS is bounded by $2|\mathcal{V}_x|$. When a type 1 or a type 3 node x is explored, the total number of type 2 admissible forwarding moves that it starts is bounded by $|\mathcal{V}^x_{xspan}| \leq k$. So the total number of type 2 moves is bounded by $2 \cdot k \cdot |\mathcal{V}_x|$. The total number of labeling of transmitting nodes as unexplored type 2 nodes is therefore bounded by $2 \cdot k \cdot |\mathcal{V}_x|$. ■

Now it is easy to conclude that the total number of explorations to transmitting nodes invoked by MDFS is bounded by $O(|\mathcal{V}_x| \cdot k)$ in iteration k of our algorithm, so MDFS terminates in finite time.

Now let's consider the computational complexity of our algorithm. We already proved that in iteration k the total number of type 1, type 2 and type 3 transmitting nodes explored by

MDFS is bounded by $|\mathcal{V}_x|$, $2|\mathcal{V}_x| \cdot k$ and $|\mathcal{V}_x|$ respectively. The worst case in computation for all these transmitting nodes is that (1) MDFS computes $\mathcal{V}^x_{xspan}$ (with complexity bounded by $O(k^3)$ based on Lemma 1) and finds the paths $P_{x\rightarrow\mathcal{V}^x_{xspan}}$ (with complexity bounded by $O(k^2)$ based on Lemma 2) for any x being explored as type 1 or type 3 node and (2) MDFS checks the rank of the binary adjacency matrix associated with each incident edge to x (with complexity bounded by $O(k)$ based on Lemma 5) for any x being explored as type 1, type 2 or type 3 node. Clearly, the total number of such edges is bounded by $2dk|\mathcal{V}_x|$ in iteration k. So the computational complexity of MDFS in iteration k is bounded by $O(|\mathcal{V}_x| \cdot k^3 + d \cdot |\mathcal{V}_x| \cdot k^2)$. The total computational complexity of our algorithm is bounded by $O(|\mathcal{V}_x| \cdot C^4 + d \cdot |\mathcal{V}_x| \cdot C^3)$ if our algorithm stops after finding C linearly independent $S - D$ paths. ■

Our algorithm shows a significant improvement in terms of computational complexity over the algorithms for solving the same problem in [3] by Amaudruz and Fragouli (whose complexity is bounded by $O(M \cdot |\mathcal{E}| \cdot C^5)$) and over the algorithm in [4] by Yazdi and Savari (whose complexity is bounded by $O(L^8 \cdot M^{12} \cdot h_0^3 + L \cdot M^6 \cdot C \cdot h_0^4)$). Note that here $M \geq d$ (because each transmitting node can transmit at most one bit information to each super node by definition of the deterministic channel model), $|\mathcal{E}| \geq |\mathcal{V}_x|$ (because of broadcasting) and $h_0 \geq C$ (because C cannot be larger than the maximum number of transmitting nodes among all layers).

B. Proof of Correctness

Theorem 2: Our algorithm finds C linearly independent paths in a linear layered deterministic relay network $\mathcal{G}$ where C is the unicast capacity (or the minimum cut value among all cuts separating the source from the destination) of $\mathcal{G}$.

Proof: We prove Theorem 2 by proving that when our algorithm stops the number of paths we find equals some cut value in $\mathcal{G}$.

If MDFS returns Res=False in iteration $K+1$ of our algorithm, then it finds the maximum number K of linearly independent $S - D$ paths in $\mathcal{G}$ and we claim $K = C$. Consider the cut Ω_K separating the super nodes labeled explored from the super nodes labeled unexplored in iteration $K+1$ when the algorithm stops so that $S \in \Omega_K$. Clearly Ω_K is a cut separating S from D and $S \in \Omega_K$, $D \in \Omega_K^c$. We prove Theorem 2 by proving that this cut value equals K, i.e, rank$(T(\mathcal{E}_{\Omega_K})) = K$.

Let $\mathcal{E}^i_{\mathcal{P}} = \{(x,y) : (x,y) \in \mathcal{E}_{\mathcal{P}}$ and $\mathcal{L}(x) = i\}$. Let $\mathcal{V}^i_{x\mathcal{P}} = \{x : (x,y) \in \mathcal{E}^i_{\mathcal{P}}\}$ and $\mathcal{V}^i_{y\mathcal{P}} = \{y :$

$(x,y) \in \mathcal{E}^i_{\mathcal{P}}\}$. We divide the set $\mathcal{E}^i_{\mathcal{P}}$ into four subgroups: $\mathcal{E}^i_{\mathcal{P}1} = \{(x,y) : (x,y) \in \mathcal{E}^i_{\mathcal{P}}, \mathcal{N}(x) \in \Omega_K, \mathcal{N}(y) \in \Omega^c_K\}$, $\mathcal{E}^i_{\mathcal{P}2} = \{(x,y) : (x,y) \in \mathcal{E}^i_{\mathcal{P}}, \mathcal{N}(x) \in \Omega_K, \mathcal{N}(y) \in \Omega_K\}$, $\mathcal{E}^i_{\mathcal{P}3} = \{(x,y) : (x,y) \in \mathcal{E}^i_{\mathcal{P}}, \mathcal{N}(x) \in \Omega^c_K, \mathcal{N}(y) \in \Omega^c_K\}$ and $\mathcal{E}^i_{\mathcal{P}4} = \{(x,y) : (x,y) \in \mathcal{E}^i_{\mathcal{P}}, \mathcal{N}(x) \in \Omega^c_K, \mathcal{N}(y) \in \Omega_K\}$. We divide the sets $\mathcal{V}^i_{x\mathcal{P}}$ ($\mathcal{V}^i_{y\mathcal{P}}$) into four subgroups accordingly, $\mathcal{V}^i_{x\mathcal{P}j}$ ($\mathcal{V}^i_{y\mathcal{P}j}$), $1 \leq j \leq 4$. Clearly, the subgroups $\mathcal{V}^i_{x\mathcal{P}j}, 1 \leq j \leq 4$ are disjoint, so are true for subgroups $\mathcal{V}^i_{y\mathcal{P}j}, 1 \leq j \leq 4$. Denote $|\mathcal{E}^i_{\mathcal{P}1}| = |\mathcal{V}^i_{x\mathcal{P}1}| = |\mathcal{V}^i_{y\mathcal{P}1}| = K_i$. Clearly K_i is the number of our identified paths (or path edges) that cross the cut Ω_K in layer cut i. Denote $|\mathcal{E}^i_{\mathcal{P}2}| = K_{i2}$, $|\mathcal{E}^i_{\mathcal{P}3}| = K_{i3}$ and $|\mathcal{E}^i_{\mathcal{P}4}| = K_{i4}$. Clearly, $K_i + K_{i2} + K_{i3} + K_{i4} = K$.

Let $\mathcal{E}^i_{\Omega_K} = \{(x,y) : (x,y) \in \mathcal{E}, \mathcal{N}(x) \in \Omega_K, \mathcal{N}(y) \in \Omega^c_K \text{ and } \mathcal{L}(x) = i\}, 1 \leq i < L$, i.e., $\mathcal{E}^i_{\Omega_K}$ is the intersection of the cut Ω_K and layer cut i. Consider the adjacency matrix $T(\mathcal{E}_{\Omega_K})$ for Ω_K. It is a block diagonal matrix with each block $T^i_{\Omega_K}$ being the adjacency matrix for $\mathcal{E}^i_{\Omega_K}$ and $\text{rank}(T(\mathcal{E}_{\Omega_K})) = \sum_{i=1}^{L-1} \text{rank}(T^i_{\Omega_K})$. Let $\mathcal{V}^i_{x\Omega_K} = \{x : (x,y) \in \mathcal{E}^i_{\Omega_K}\}$ and $\mathcal{V}^i_{y\Omega_K} = \{y : (x,y) \in \mathcal{E}^i_{\Omega_K}\}$. We also divide the set $\mathcal{V}^i_{x\Omega_K}$ into three subgroups: $\mathcal{V}^i_{x\Omega_K 1} = \mathcal{V}^i_{x\mathcal{P}1}$, $\mathcal{V}^i_{x\Omega_K 2} = \mathcal{V}^i_{x\mathcal{P}2}$ and $\mathcal{V}^i_{x\Omega_K 3} = \mathcal{V}^i_{x\Omega_K} - \mathcal{V}^i_{x\Omega_K 1} - \mathcal{V}^i_{x\Omega_K 2}$. Similarly, we divide the set $\mathcal{V}^i_{y\Omega_K}$ into three subgroups: $\mathcal{V}^i_{y\Omega_K 1} = \mathcal{V}^i_{y\mathcal{P}1}$, $\mathcal{V}^i_{y\Omega_K 2} = \mathcal{V}^i_{y\mathcal{P}3}$ and $\mathcal{V}^i_{y\Omega_K 3} = \mathcal{V}^i_{y\Omega_K} - \mathcal{V}^i_{y\Omega_K 1} - \mathcal{V}^i_{y\Omega_K 2}$. Clearly, the subgroups $\mathcal{V}^i_{x\Omega_K j}, 1 \leq j \leq 3$ are disjoint, so are true for subgroups $\mathcal{V}^i_{y\Omega_K j}, 1 \leq j \leq 3$.

Denote $\mathcal{E}^i_{u,x_k}$, $\mathcal{V}^i_{xu,x_k}$ and $\mathcal{V}^i_{yu,x_k}$ as the instantaneous sets of $\mathcal{E}^i_u$, $\mathcal{V}^i_{xu}$ and $\mathcal{V}^i_{yu}$ when a transmitting node x_k is being explored with $i = \mathcal{L}(x_k)$ in iteration $K+1$ of our algorithm. We divide the set $\mathcal{E}^i_{u,x_k}$ into four subgroups as follows. Let $\mathcal{E}^i_{u,x_k,1} = \{(x,y) \in \mathcal{E}^i_{u,x_k}, \mathcal{N}(x) \in \Omega_K, \mathcal{N}(y) \in \Omega^c_K\}$, $\mathcal{E}^i_{u,x_k,2} = \{(x,y) \in \mathcal{E}^i_{u,x_k}, \mathcal{N}(x) \in \Omega_K, \mathcal{N}(y) \in \Omega_K\}$, $\mathcal{E}^i_{u,x_k,3} = \{(x,y) \in \mathcal{E}^i_{u,x_k}, \mathcal{N}(x) \in \Omega^c_K, \mathcal{N}(y) \in \Omega^c_K\}$ and $\mathcal{E}^i_{u,x_k,4} = \{(x,y) \in \mathcal{E}^i_{u,x_k}, \mathcal{N}(x) \in \Omega^c_K, \mathcal{N}(y) \in \Omega_K\}$. Divide $\mathcal{V}^i_{xu,x_k}$ and $\mathcal{V}^i_{yu,x_k}$ respectively into four subgroups $\mathcal{V}^i_{xu,x_k,j}$ and $\mathcal{V}^i_{yu,x_k,j}$ corresponding to $\mathcal{E}^i_{u,x_k,j}, 1 \leq j \leq 4$. Clearly $|\mathcal{E}^i_{u,x_k,1}| + |\mathcal{E}^i_{u,x_k,2}| + |\mathcal{E}^i_{u,x_k,3}| + |\mathcal{E}^i_{u,x_k,4}| = K$ and $\text{rank}(T(\mathcal{E}^i_{u,x_k})) = K$ for any x_k ever explored by MDFS in iteration $K+1$ of our algorithm based on Lemma 3.

In the following of this section, we prove a sequence of lemmas before we finally prove Theorem 2. Unless otherwise stated, we assume that we are in the last iteration $K+1$ of our algorithm.

Lemma 7: Let x_k be a transmitting node that has been explored by MDFS in iteration $K+1$

of our algorithm with $\mathcal{L}(x_k) = i$. When x_k is explored, we have $\mathcal{V}^{x_k}_{xspan} \subseteq \mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}$ and

$$\text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k})) \tag{2}$$

$$= \text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k})) = |\mathcal{V}^i_{xu,x_k,1}| + |\mathcal{V}^i_{xu,x_k,2}| \tag{3}$$

For any $y \in \mathcal{V}^i_{y\Omega_K 3}$, we have

$$\text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k} + y)) \tag{4}$$

$$= \text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k} + y)) = |\mathcal{V}^i_{xu,x_k,1}| + |\mathcal{V}^i_{xu,x_k,2}| \tag{5}$$

Proof: Based on Lemma 6, all $\mathcal{N}(x_j)$ with $x_j \in \mathcal{V}^{x_k}_{xspan}$ will finally be explored by MDFS in iteration $K+1$ if x_k has ever been explored. Since for any $x \in \mathcal{V}^i_{xu,x_k,3} + \mathcal{V}^i_{xu,x_k,4}$, $\mathcal{N}(x)$ is not explored by MDFS, so we have $\mathcal{V}^{x_k}_{xspan} \subseteq \mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}$. By definition of $\mathcal{V}^{x_k}_{xspan}$, it's easy to conclude that (2)-(3) hold. By definition $\mathcal{N}(y)$ is not explored by MDFS for any $y \in \mathcal{V}^i_{y\Omega_K 3}$. Based on Lemma 6, we have $\text{rank}(T(\mathcal{V}^i_{xu,x_k} + x_k, \mathcal{V}^i_{yu,x_k} + y)) = K$. Given $\text{rank}(T(\mathcal{V}^i_{xu,x_k}, \mathcal{V}^i_{yu,x_k} + y)) = K$ and the fact that $\mathcal{V}^{x_k}_{xspan} \subseteq \mathcal{V}^i_{xu,x_k}$ is the unique set satisfying $\text{rank}(T(\mathcal{V}^i_{xu,x_k} + x_k, \mathcal{V}^i_{yu,x_k})) = \text{rank}(T(\mathcal{V}^i_{xu,x_k}, \mathcal{V}^i_{yu,x_k})) = K$, we conclude $\text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k} + y)) = \text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k} + y)) = |\mathcal{V}^i_{xu,x_k,1}| + |\mathcal{V}^i_{xu,x_k,2}|$. ■

Lemma 8: For any x_k with $\mathcal{L}(x_k) = i$ explored by MDFS in iteration $K+1$ of our algorithm, $|\mathcal{E}^i_{u,x_k,1}| + |\mathcal{E}^i_{u,x_k,3}| = |\mathcal{E}^i_{\mathcal{P}1}| + |\mathcal{E}^i_{\mathcal{P}3}| = K_i + K_{i3}$, $|\mathcal{E}^i_{u,x_k,3}| + |\mathcal{E}^i_{u,x_k,4}| = |\mathcal{E}^i_{\mathcal{P}3}| + |\mathcal{E}^i_{\mathcal{P}4}| = K_{i3} + K_{i4}$ and $|\mathcal{E}^i_{u,x_k,1}| - |\mathcal{E}^i_{u,x_k,4}| = |\mathcal{E}^i_{\mathcal{P}1}| - |\mathcal{E}^i_{\mathcal{P}4}| = K_i - K_{i4}$.

Proof: By definition all the super nodes where the nodes in $\mathcal{V}^i_{x\mathcal{P}3}$, $\mathcal{V}^i_{x\mathcal{P}4}$, $\mathcal{V}^i_{y\mathcal{P}1}$ and $\mathcal{V}^i_{y\mathcal{P}3}$ belong to are labeled unexplored when MDFS returns, so during the running time of MDFS in iteration $K+1$, the nodes in $\mathcal{V}^i_{x\mathcal{P}3}$, $\mathcal{V}^i_{x\mathcal{P}4}$, $\mathcal{V}^i_{y\mathcal{P}1}$ and $\mathcal{V}^i_{y\mathcal{P}3}$ are never explored and $\mathcal{V}^i_{x\mathcal{P}3} \subseteq \mathcal{V}^i_{xu,x_k}$, $\mathcal{V}^i_{x\mathcal{P}4} \subseteq \mathcal{V}^i_{xu,x_k}$, $\mathcal{V}^i_{y\mathcal{P}1} \subseteq \mathcal{V}^i_{yu,x_k}$ and $\mathcal{V}^i_{y\mathcal{P}3} \subseteq \mathcal{V}^i_{yu,x_k}$ always hold for any x_k with $\mathcal{L}(x_k) = i$ explored by MDFS. Any $\mathcal{N}(x)$ (or $\mathcal{N}(y)$) with transmitting node x (or receiving node y) used by $\mathcal{E}^i_u$ but not by $\mathcal{E}^i_{\mathcal{P}}$ must be explored by MDFS, which means that for any x_k with $\mathcal{L}(x_k) = i$ explored by MDFS $\mathcal{V}^i_{x\mathcal{P}34} = \mathcal{V}^i_{x\mathcal{P}3} + \mathcal{V}^i_{x\mathcal{P}4}$ is the complete set of transmitting nodes in $\mathcal{V}^i_{xu,x_k}$ so that for each transmitting node x in this set, $\mathcal{N}(x)$ is unexplored and $\mathcal{V}^i_{y\mathcal{P}13} = \mathcal{V}^i_{y\mathcal{P}1} + \mathcal{V}^i_{y\mathcal{P}3}$ is the complete set of receiving nodes in $\mathcal{V}^i_{yu,x_k}$ so that for each receiving node y in this set, $\mathcal{N}(y)$ is unexplored. So $|\mathcal{E}^i_{u,x_k,1}| + |\mathcal{E}^i_{u,x_k,3}| = |\mathcal{E}^i_{\mathcal{P}1}| + |\mathcal{E}^i_{\mathcal{P}3}| = K_i + K_{i3}$ and $|\mathcal{E}^i_{u,x_k,3}| + |\mathcal{E}^i_{u,x_k,4}| = |\mathcal{E}^i_{\mathcal{P}3}| + |\mathcal{E}^i_{\mathcal{P}4}| = K_{i3} + K_{i4}$. $|\mathcal{E}^i_{u,x_k,1}| - |\mathcal{E}^i_{u,x_k,4}| = |\mathcal{E}^i_{\mathcal{P}1}| - |\mathcal{E}^i_{\mathcal{P}4}| = K_i - K_{i4}$ is a straightforward result by subtracting one from the other. ■

Lemma 9: rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))\geq|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$ for x_k being explored by MDFS.

Proof: If $K_i-K_{i4}<0$, the statement is obviously true. Assume $K_i-K_{i4}\geq 0$. We know rank$(T(\mathcal{E}^i_{u,x_k}))=K=|\mathcal{E}^i_{u,x_k,1}|+|\mathcal{E}^i_{u,x_k,2}|+|\mathcal{E}^i_{u,x_k,3}|+|\mathcal{E}^i_{u,x_k,4}|=$ rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+\mathcal{V}^i_{xu,x_k,3}+\mathcal{V}^i_{xu,x_k,4},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,3}+\mathcal{V}^i_{yu,x_k,4}))$. If rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))<|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|$, we will have rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+\mathcal{V}^i_{xu,x_k,3}+\mathcal{V}^i_{xu,x_k,4},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,3}+\mathcal{V}^i_{yu,x_k,4}))<|\mathcal{E}^i_{u,x_k,1}|+|\mathcal{E}^i_{u,x_k,2}|+|\mathcal{E}^i_{u,x_k,3}|+|\mathcal{E}^i_{u,x_k,4}|=K$. ■

Let x_k be some transmitting node explored by MDFS in iteration $K+1$ of our algorithm. Let $y'\in\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,4}$. By definition, $\mathcal{N}(y')$ is explored by MDFS sometime. If $y'\in\mathcal{V}^i_{y\mathcal{P}}$, then y' must have been deleted from $\mathcal{V}^i_{yu}$ in a type 3 forwarding move of MDFS sometime in iteration $K+1$ of our algorithm given that $\mathcal{N}(y')$ is explored and have already been added back to $\mathcal{V}^i_{yu,x_k}$ if this type 3 forwarding move on y' happens before the current exploration of x_k otherwise it won't appear in $\mathcal{V}^i_{yu,x_k}$. If $y'\notin\mathcal{V}^i_{y\mathcal{P}}$, then y' must have been added to $\mathcal{V}^i_{yu,x_k}$ in a type 1 forwarding move by MDFS before the current exploration to x_k otherwise y' won't appear in $\mathcal{V}^i_{yu,x_k}$ when x_k is explored. It means for each $y'\in\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,4}$, y' is either added to $\mathcal{V}^i_{yu,x_k}$ before the current exploration of x_k or is deleted from $\mathcal{V}^i_{yu,x_k}$ after the current exploration of x_k.

Lemma 10: If rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))>|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$ for x_k being explored by MDFS, then there exists some nonempty set $\mathcal{V}_{y24,x_k}\subseteq\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,4}$, such that for any $y''\in\mathcal{V}_{y24,x_k}$, $T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y'')=\sum_{y_i\in\mathcal{V}''_y}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)$ for some $\mathcal{V}''_y=\mathcal{V}_{y24,x_k}-y''+\mathcal{V}''_{y1}$ with $\mathcal{V}''_{y1}\subseteq\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}$. Let $\mathcal{E}_{u24,x_k}\subseteq\mathcal{E}^i_{u,x_k,2}+\mathcal{E}^i_{u,x_k,4}$ be the edge set with $\mathcal{V}_{y24,x_k}$ being their receiving nodes.

Proof: We know that rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,3}+\mathcal{V}^i_{yu,x_k,4}))=|\mathcal{V}^i_{xu,x_k,1}|+|\mathcal{V}^i_{xu,x_k,2}|$. If rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))>|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|$, there must exist some $y'\in\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,4}$, such that $T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y')=\sum_{y_i\in\mathcal{V}'_y}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)$ for some $\mathcal{V}'_y\subseteq\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,3}+\mathcal{V}^i_{yu,x_k,4}-y'$. Let $\mathcal{V}_{y24,x_k}=\{y'+\mathcal{V}'_y\}\cap\{\mathcal{V}^i_{yu,x_k,2}+\mathcal{V}^i_{yu,x_k,4}\}$. Obviously $\mathcal{V}_{y24,x_k}\neq\emptyset$ and $\sum_{y_i\in\mathcal{V}_{y24,x_k}}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)=\sum_{y_i\in\mathcal{V}'_y-\mathcal{V}_{y24,x_k}}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)$. For any $y''\in\mathcal{V}_{y24,x_k}$, we have $T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y'')=\sum_{y_i\in\mathcal{V}_{y24,x_k}-y''}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)+\sum_{y_i\in\mathcal{V}''_{y1}}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)=\sum_{y_i\in\mathcal{V}''_y}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i)$ with $\mathcal{V}''_{y1}=\mathcal{V}'_y-\mathcal{V}_{y24,x_k}\subseteq\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}$ and $\mathcal{V}''_y=\mathcal{V}_{y24,x_k}-y''+\mathcal{V}''_{y1}$. ■

Assume rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))>|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$. Let $y''\in\mathcal{V}_{y24,x_k}$ be the last one in $\mathcal{V}_{y24,x_k}$ being added to the set $\mathcal{V}^i_{yu,x_k}$ before the current exploration

of x_k or the first one in $\mathcal{V}_{y24,x_k}$ being deleted from the set $\mathcal{V}^i_{yu,x_k}$ after the current exploration of x_k. By definition,

$$T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y'')=\sum_{y_i\in\mathcal{V}''_y}T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_i) \tag{6}$$

for some $\mathcal{V}''_y=\mathcal{V}_{y24,x_k}-y''+\mathcal{V}''_{y1}\subseteq\mathcal{V}^i_{yu,x_k}-y''$ with $\mathcal{V}''_{y1}\subseteq\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}$. Let x'' be the corresponding transmitting node such that $(x'',y'')\in\mathcal{E}_{u24,x_k}$.

Lemma 11: Assume rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))>|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$ and let $(x'',y'')\in\mathcal{E}_{u24,x_k}$ be defined above. Then just after adding y'' or just before deleting y'', we have

$$T(\mathcal{V}^i_{xu,x'',1}+\mathcal{V}^i_{xu,x'',2}+x'',y'')=\sum_{y_i\in\mathcal{V}''_y}T(\mathcal{V}^i_{xu,x'',1}+\mathcal{V}^i_{xu,x'',2}+x'',y_i) \tag{7}$$

for the same $\mathcal{V}''_y$ as in Equation (6). And when x'' is explored just before adding y'' along edge (x'',y'') or just after deleting y'' along edge (x'',y''), we have rank$(T(\mathcal{V}^i_{xu,x'',1}+\mathcal{V}^i_{xu,x'',2}+x'',\mathcal{V}^i_{yu,x''}))=|\mathcal{V}^i_{xu,x'',1}|+|\mathcal{V}^i_{xu,x'',2}|+1$.

Proof: Note that in equation (7), since $y''\in\mathcal{V}_{y24,x_k}$ with $(x'',y'')\in\mathcal{E}^i_{u,x_k}$ is the last one in the set $\mathcal{V}_{y24,x_k}$ being added to the set $\mathcal{V}^i_{yu,x_k}$ before the current exploration of x_k or the first one in the set $\mathcal{V}_{y24,x_k}$ being deleted from the set $\mathcal{V}^i_{yu,x_k}$ after the current exploration of x_k, $\mathcal{V}_{y24,x_k}-y''$ is not changed and is also subset of $\mathcal{V}^i_{yu,x''}$. So in equation (7), $\mathcal{V}''_y$ is the same as in (6) given that $\mathcal{V}_{y24,x_k}-y''$ is not changed and $\mathcal{V}''_{y1}\subseteq\mathcal{V}^i_{yu,x_i,1}+\mathcal{V}^i_{yu,x_i,3}=\mathcal{V}^i_{y\mathcal{P}1}+\mathcal{V}^i_{y\mathcal{P}3}$ is not changed, but the set $\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}$ are changing to $\mathcal{V}^i_{xu,x'',1}+\mathcal{V}^i_{xu,x'',2}$. When MDFS proceeds from the point of just after adding (x'',y'') to $\mathcal{E}_u$ to the point of exploring x_k or from the point of exploring x_k to the point of just before deleting (x'',y'') from $\mathcal{E}_u$, only those three forwarding moves in definition of MDFS are allowed. It is sufficient for us to show that any forwarding moves of MDFS or backtracking of these moves doesn't change the relationship in equation (6) so that equation (7) holds.

Let's first consider forwarding moves of MDFS. A type 1 forwarding move along edge (x,y) would change $\mathcal{V}^i_{xu,x}$ to $\mathcal{V}^i_{xu,x}+x$. Since $\mathcal{V}^x_{xspan}\subseteq\mathcal{V}^i_{xu,x,1}+\mathcal{V}^i_{xu,x,2}$ (based on Lemma 7), the vector $T(x,\mathcal{V}^i_{yu,x})$ is a linear combination of row vectors in $T(\mathcal{V}^i_{xu,x,1}+\mathcal{V}^i_{xu,x,2},\mathcal{V}^i_{yu,x})$, so the relationship in (6) still holds while $\mathcal{V}^i_{xu,x,1}+\mathcal{V}^i_{xu,x,2}$ changes to $\mathcal{V}^i_{xu,x,1}+\mathcal{V}^i_{xu,x,2}+x$ in a type 1 forwarding move. In a type 2 forwarding move, $\mathcal{V}^i_{xu,x,1}+\mathcal{V}^i_{xu,x,2}$ changes to $\mathcal{V}^i_{xu,x,1}+\mathcal{V}^i_{xu,x,2}+x-x_i$ for some

$x_i \in \mathcal{V}^x_{xspan}$. Again we have $\mathcal{V}^x_{xspan} \subseteq \mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ (based on Lemma 7). It is easy to see that the relationship in (6) still holds when $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ changes to $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2} + x - x_i$. In a type 3 forwarding move, some $y \in \mathcal{V}^i_{yu,x} \backslash \mathcal{V}_{y24,x_k}$ is deleted from $\mathcal{V}^i_{yu,x}$ which obviously doesn't affect the relationship in (6). Now let's consider backtracking moves of MDFS. Let equation (6) hold after a type 3 forwarding move of MDFS along an edge (x, y) for some $y \notin \mathcal{V}_{y24,x_k}$. After the type 3 forwarding move along (x, y), MDFS will explore x when it explores $\mathcal{N}(x)$. Again we have $\mathcal{V}^x_{xspan} \subseteq \mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$, so equation (6) should hold before the type 3 forwarding move when $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ was $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2} + x$. A proceeding type 2 forwarding move before the current exploration of x means $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ was $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2} + x - x'$ with $x \in \mathcal{V}^{x'}_{xspan}$ before the move. Let equation (6) hold after a type 2 forwarding move of MDFS along a path $P_{x' \to x}$ with $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$. Again we have $\mathcal{V}^x_{xspan} \subseteq \mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ (based on Lemma 7), so equation (6) holds with addition of row for x, which means that equation (6) holds before the type 2 forwarding move when $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ was $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2} + x - x'$. If the proceeding move is a type 1 forwarding move, it means that $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$ was $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2} - x$, it is obvious that equation (6) holds before the type 1 forwarding move with $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2} - x$ if it holds after the move with $\mathcal{V}^i_{xu,x,1} + \mathcal{V}^i_{xu,x,2}$. From above discussion, we conclude that equation (7) holds.

We know that after adding (x'', y'') or before deleting (x'', y''), rank$(T(\mathcal{V}^i_{xu,x'',1} + \mathcal{V}^i_{xu,x'',2} + x'', \mathcal{V}^i_{yu,x''} + y'')) = |\mathcal{V}^i_{xu,x'',1}| + |\mathcal{V}^i_{xu,x'',2}| + 1$. Given that equation (7) holds, we have rank$(T(\mathcal{V}^i_{xu,x'',1} + \mathcal{V}^i_{xu,x'',2} + x'', \mathcal{V}^i_{yu,x''})) = |\mathcal{V}^i_{xu,x'',1}| + |\mathcal{V}^i_{xu,x'',2}| + 1$. ■

Lemma 12: For any x_k that has been explored by MDFS in iteration $K+1$ of our algorithm, rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3})) = |\mathcal{V}^i_{xu,x_k,1}| - |\mathcal{V}^i_{xu,x_k,4}| = K_i - K_{i4} \geq 0$.

Proof: We first prove $K_i - K_{i4} \geq 0$. Assume $K_i - K_{i4} < 0$. Then we must have rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3})) > |\mathcal{V}^i_{xu,x_k,1}| - |\mathcal{V}^i_{xu,x_k,4}| = K_i - K_{i4}$. Based on Lemma 10 and 11, if rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3})) > |\mathcal{V}^i_{xu,x_k,1}| - |\mathcal{V}^i_{xu,x_k,4}| = K_i - K_{i4}$, then we will have rank$(T(\mathcal{V}^i_{xu,x'',1} + \mathcal{V}^i_{xu,x'',2} + x'', \mathcal{V}^i_{yu,x''})) = |\mathcal{V}^i_{xu,x'',1}| + |\mathcal{V}^i_{xu,x'',2}| + 1$ when x'' is explored just before adding y'' along edge (x'', y'') or just after deleting y'' along edge (x'', y'') for x'', y'' defined as in Lemma 11. But based on Lemma 7, we should have rank$(T(\mathcal{V}^i_{xu,x'',1} + \mathcal{V}^i_{xu,x'',2} + x'', \mathcal{V}^i_{yu,x''})) = |\mathcal{V}^i_{xu,x'',1}| + |\mathcal{V}^i_{xu,x'',2}|$, which constitutes a contradiction. So we must have $K_i - K_{i4} \geq 0$.

Assume rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3})) > |\mathcal{V}^i_{xu,x_k,1}| - |\mathcal{V}^i_{xu,x_k,4}| = K_i - K_{i4} \geq$

0. Using a similar argument as above, we would arrive at a contradiction which means the assumption doesn't hold. So we must have rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))\leq |\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$. Now together with Lemma 9, we conclude that rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$. ■

Lemma 13: For any x_k with $\mathcal{L}(x_k)=i$ that has been explored by MDFS in iteration $K+1$ of our algorithm and any $y_j\in\mathcal{V}^i_{y\Omega_K 3}$, rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}+y_j))$ =rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$.

Proof: First we prove rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))=K_i-K_{i4}$. Based on Lemma 7, $\mathcal{V}^{x_k}_{xspan}\subseteq\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}$, so rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))$ =rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$ based on Lemma 12.

Second we prove rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}+y_j))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$. Given rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|=K_i-K_{i4}$, rank$(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}+y_j))$ equals either $|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|$ or $|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|+1$. Assume

$$\text{rank}(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}+y_j))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|+1 \tag{8}$$

From Lemma 7, we have

$$\text{rank}(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}+x_k,\mathcal{V}^i_{yu,x_k}+y_j)) \tag{9}$$

$$=\ \text{rank}(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k}+y_j))=|\mathcal{V}^i_{xu,x_k,1}|+|\mathcal{V}^i_{xu,x_k,2}| \tag{10}$$

and

$$T(x_k,\mathcal{V}^i_{yu,x_k}+y_j) \tag{11}$$

$$=\sum_{x'\in\mathcal{V}^{x_k}_{xspan}\subseteq\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2}} T(x',\mathcal{V}^i_{yu,x_k}+y_j) \tag{12}$$

From (8) and (12), we have

$$\text{rank}(T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},\mathcal{V}^i_{yu,x_k,1}+\mathcal{V}^i_{yu,x_k,3}+y_j))=|\mathcal{V}^i_{xu,x_k,1}|-|\mathcal{V}^i_{xu,x_k,4}|+1 \tag{13}$$

From (10) and (13), we have

$$T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_j) \tag{14}$$

$$=\sum_{y_{13}\in\mathcal{V}_{y13,x_k}} T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_{13})+\sum_{y_{24}\in\mathcal{V}_{y24,x_k}} T(\mathcal{V}^i_{xu,x_k,1}+\mathcal{V}^i_{xu,x_k,2},y_{24}) \tag{15}$$

for some $\mathcal{V}_{y13,x_k} \subseteq \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3}$ and some nonempty $\mathcal{V}_{y24,x_k} \subseteq \mathcal{V}^i_{yu,x_k,2} + \mathcal{V}^i_{yu,x_k,4}$. From (12) and (15), we have

$$T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, y_j) \tag{16}$$

$$= \sum_{y_{13} \in \mathcal{V}_{y13,x_k}} T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, y_{13}) + \sum_{y_{24} \in \mathcal{V}_{y24,x_k}} T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, y_{24}) \tag{17}$$

for the same $\mathcal{V}_{y13,x_k}$ and $\mathcal{V}_{y24,x_k}$ in (15).

Let $y'' \in \mathcal{V}_{y24,x_k}$ be the last one in $\mathcal{V}_{y24,x_k}$ being added to the set $\mathcal{V}^i_{yu,x_k}$ before the current exploration of x_k or the first one in $\mathcal{V}_{y24,x_k}$ being deleted from the set $\mathcal{V}^i_{yu,x_k}$ after the current exploration of x_k and $(x'', y'') \in \mathcal{E}^i_{u,x_k}$. Then using a similar argument as that in Lemma 11, we have rank$(T(\mathcal{V}^i_{xu,x'',1} + \mathcal{V}^i_{xu,x'',2} + x'', \mathcal{V}^i_{yu,x''} + y_j)) = |\mathcal{V}^i_{xu,x'',1}| + |\mathcal{V}^i_{xu,x'',2}| + 1$ when x'' is explored just before adding y'' along edge (x'', y'') or just after deleting y'' along edge (x'', y''), but it's a contradiction with Lemma 7. So it must be rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3} + y_j)) = |\mathcal{V}^i_{xu,x_k,1}| - |\mathcal{V}^i_{xu,x_k,4}| = K_i - K_{i4}$. ■

Lemma 14: rank$(T^i_{\Omega_K})$ =rank$(T(\mathcal{V}^i_{x\Omega_K}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4}, 1 \le i \le L-1$.

Proof: If $\mathcal{E}^i_{\Omega_K} = \emptyset$, i.e., the cut Ω_K and layer cut i has no intersection, then rank$(T^i_{\Omega_K})$ = rank$(T(\mathcal{V}^i_{x\Omega_K}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4} = 0$ holds. Next assume that $\mathcal{E}^i_{\Omega_K} \neq \emptyset$. Lemma 13 says for any x_k with $\mathcal{L}(x_k) = i$ that has been explored by MDFS in iteration $K+1$ of our algorithm and any $y_j \in \mathcal{V}^i_{y\Omega_K 3}$, rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3} + y_j)) = K_i - K_{i4}$ and rank$(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3})) = K_i - K_{i4}$. Lemma 7 says for any $y_j \in \mathcal{V}^i_{y\Omega_K 3}$, $T(x_k, \mathcal{V}^i_{yu,x_k} + y_j) = \sum_{x' \in \mathcal{V}^{x_k}_{xspan} \subseteq \mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}} T(x', \mathcal{V}^i_{yu,x_k} + y_j)$. Based on these two Lemmas, it's easy to conclude that

$$\text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{yu,x_k,1} + \mathcal{V}^i_{yu,x_k,3} + \mathcal{V}^i_{y\Omega_K 3})) \tag{18}$$

$$= \text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2} + x_k, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4} \tag{19}$$

$$= \text{rank}(T(\mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4} \tag{20}$$

and

$$T(x_k, \mathcal{V}^i_{y\Omega_K}) = \sum_{x' \in \mathcal{V}^{x_k}_{xspan} \subseteq \mathcal{V}^i_{xu,x_k,1} + \mathcal{V}^i_{xu,x_k,2}} T(x', \mathcal{V}^i_{y\Omega_K}) \tag{21}$$

Equations (19) (20) and (21) hold for any x_k that has been explored by MDFS in iteration $K+1$ of our algorithm.

Let $x^q, 1 \leq q \leq Q$ be the qth transmitting nodes in layer i that has been explored by MDFS in iteration $K+1$ in our algorithm. Note that since some transmitting nodes may be explored more than once, x^q may not be distinct but Q is finite. We claim that

$$\text{rank}(T(\mathcal{V}^i_{x\mathcal{P}1} + \mathcal{V}^i_{x\mathcal{P}2} + \sum_{k=1}^{q} x^k, \mathcal{V}^i_{y\Omega_K})) = \text{rank}(T(\mathcal{V}^i_{x\mathcal{P}1} + \mathcal{V}^i_{x\mathcal{P}2}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4} \tag{22}$$

for $1 \leq q \leq Q$. When x^1 is explored, $\mathcal{V}^i_{xu,x^1,1} = \mathcal{V}^i_{x\mathcal{P}1}$ and $\mathcal{V}^i_{xu,x^1,2} = \mathcal{V}^i_{x\mathcal{P}2}$. From (19) (20), we have

$$\text{rank}(T(\mathcal{V}^i_{x\mathcal{P}1} + \mathcal{V}^i_{x\mathcal{P}2} + x^1, \mathcal{V}^i_{y\Omega_K})) = \text{rank}(T(\mathcal{V}^i_{x\mathcal{P}1} + \mathcal{V}^i_{x\mathcal{P}2}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4} \tag{23}$$

When x^2 is explored,

$$\mathcal{V}^i_{xu,x^2,1} + \mathcal{V}^i_{xu,x^2,2} \subseteq \mathcal{V}^i_{xu,x^1,1} + \mathcal{V}^i_{xu,x^1,2} + x^1 \tag{24}$$

and from (19) (20)

$$\text{rank}(T(\mathcal{V}^i_{xu,x^2,1} + \mathcal{V}^i_{xu,x^2,2} + x^2, \mathcal{V}^i_{y\Omega_K})) = \text{rank}(T(\mathcal{V}^i_{xu,x^2,1} + \mathcal{V}^i_{xu,x^2,2}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4} \tag{25}$$

From (23) to (25), we conclude that $\text{rank}(T(\mathcal{V}^i_{x\mathcal{P}1} + \mathcal{V}^i_{x\mathcal{P}2} + x^1 + x^2, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4}$. Use induction on $x^q, 1 \leq q \leq Q$, just as we did for x^2, we conclude that (22) holds for any q,$1 \leq q \leq Q$. We know that $\mathcal{V}^i_{x\mathcal{P}1} + \mathcal{V}^i_{x\mathcal{P}2} + \sum_{k=1}^{Q} x^k = \mathcal{V}^i_{x\Omega_K}$, so when $q = Q$ equation (22) means that $\text{rank}(T^i_{\Omega_K})$ =$\text{rank}(T(\mathcal{V}^i_{x\Omega_K}, \mathcal{V}^i_{y\Omega_K})) = K_i - K_{i4}, 1 \leq i \leq L-1$. ■

Lemma 15:

$$\sum_{i=1}^{L-1} K_i - \sum_{i=1}^{L-1} K_{i4} = K \tag{26}$$

Proof: By definition, $\sum_{i=1}^{L-1} K_i$ is the total times the paths in $\mathcal{P}$ cross the cut Ω_K. Since each path in $\mathcal{P}$ crosses Ω_K at least once and it's possible that some of these paths may cross Ω_K more than once, we have $\sum_{i=1}^{L-1} K_i \geq K$. For $\forall P \in \mathcal{P}$, let k_p be the times P goes from Ω_K to Ω_K^c and $k_P^{'}$ be the times P goes from Ω_K^c to Ω_K. Given $S \in \Omega_K$ and $D \in \Omega_K^c$, it must be true that $k_P - k_P^{'} = 1$ and $\sum_{P\in\mathcal{P}}(k_P - k_P^{'}) = |\mathcal{P}| = K$. By definition, the times that $\forall P \in \mathcal{P}$ goes from Ω_K to Ω_K^c is counted in $\sum_{i=1}^{L-1} K_i$ and the times that $\forall P \in \mathcal{P}$ goes from Ω_K^c to Ω_K is counted in $\sum_{i=1}^{L-1} K_{i4}$, i.e., $\sum_{P\in\mathcal{P}}(k_P - k_P^{'}) = \sum_{i=1}^{L-1} K_i - \sum_{i=1}^{L-1} K_{i4}$. So we have $\sum_{i=1}^{L-1} K_i - \sum_{i=1}^{L-1} K_{i4} = K$. ■

Based on Lemma 14, we have

$$\text{rank}(T(\mathcal{E}_{\Omega_K})) = \sum_{i=1}^{L-1} \text{rank}(T^i_{\Omega_K}) = \sum_{i=1}^{L-1}(K_i - K_{i4}) = \sum_{i=1}^{L-1} K_i - \sum_{i=1}^{L-1} K_{i4} \tag{27}$$

In Lemma 15, we show that $\sum_{i=1}^{L-1} K_i - \sum_{i=1}^{L-1} K_{i4} = K$ which means rank$(T(\mathcal{E}_{\Omega_K})) = K$ and this concludes our proof for Theorem 2.

■

Theorem 1 proves that our algorithm terminates in finite time. Theorem 2 proves that our algorithm finds C linearly independent $S - D$ paths ($\mathcal{P}$) where C is the unicast capacity of the underlying deterministic relay network. Lemma 4 shows that these C paths in $\mathcal{P}$ correspond to a capacity-achieving transmission scheme. They consist the complete proof of correctness for our algorithm for finding the unicast capacity of any linear layered deterministic wireless relay network.

An arbitrary deterministic relay network $\mathcal{G}$ can be unfolded over time to create a layered deterministic network $\mathcal{G}_L$ through time-expansion technique [1][2]. The transmission scheme in $\mathcal{G}_L$ identified by our algorithm corresponds to some equivalent transmission scheme in $\mathcal{G}$ maybe time-variant. It means that our algorithm works for finding the unicast capacity of an arbitrary linear deterministic relay network.

V. Conclusions

The deterministic channel model for wireless relay networks has been a useful model for studying the capacity and capacity-approaching transmission schemes for underlying networks. In this paper, we proposed a fast algorithm for finding the unicast capacity of any given linear deterministic wireless relay network. Our algorithm finds the maximum number of linearly independent paths for the deterministic relay network and these paths correspond to a capacity-achieving transmission scheme. The essential component of our algorithm is a modified depth-first search algorithm developed for linear deterministic wireless relay networks. The proof of correctness for the algorithm is given which guarantees that our algorithm works in universal cases. Compared with previous results on solving the same problem, our algorithm prevails with significantly lower computational complexity. Moreover, the development of the modified depth-first search algorithm is based on a very intuitive idea, that is, to build up the path by adding edges while avoiding the linear dependency by using rank check at the same time.

References

[1] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse, "A Deterministic Approach to Wireless Relay Networks," *Proceedings of Allerton Conference on Communication, Control and Computing, Illinois*, Sep. 2007.

[2] ——, "Wireless Network Information Flow," *Proceedings of Allerton Conference on Communication, Control and Computing, Illinois*, Sep. 2007.

[3] A. Amaudruz and C. Fragouli, "Combinatorial Algorithms for Wireless Information Flow," *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 555 – 564, Jan. 2009.

[4] S. M. S. T. Yazdi and S. A. Savari, "A Combinatorial Study of Linear Deterministic Relay Networks," *http://arxiv.org/abs/0904.2401*, Apr. 2009.

[5] http://www.ece.iastate.edu/˜cshi.